

## FROM INDEXED GRAMMARS TO GENERATING FUNCTIONS

JARED ADAMS, ERIC FREDEN, AND MARNI MISHNA

ABSTRACT. We extend the DSV method of computing the growth series of an unambiguous context-free language to the larger class of indexed languages. We illustrate the technique with numerous examples.

## 1. INTRODUCTION

**1.1. Indexed grammars.** Indexed grammars were introduced in the thesis of Aho in the late 1960s to model a more natural subclass of context-sensitive languages with interesting closure properties [1, 14], but which is more expressive than context-free grammars. The original reference for basic results on indexed grammars is [1]. The complete definition of these grammars is equivalent to the following reduced form.

**Definition.** A *reduced indexed grammar* is a 5-tuple  $(V, \Sigma, I, P, S)$ , where  $V$  is a set of variables (also referred to as non-terminals),  $\Sigma$  the set of terminals,  $I$  the set of indices,  $S \in V$  the start symbol, and  $P$  a finite set of productions of the form

$$(1.1) \quad \mathbf{A} \rightarrow \alpha \quad \mathbf{A} \rightarrow \mathbf{B}_f \quad \mathbf{A}_f \rightarrow \alpha$$

where  $\mathbf{A}, \mathbf{B} \in V$ ,  $f \in I$  and  $\alpha \in (V \cup \Sigma)^*$ . Derivations are similar to those in a context-free grammar, except that each variable is associated to a dedicated stack, which we indicate here with a subscript. The expressive power of an indexed grammar comes from the treatment of the stack. For example, in productions of the form  $\mathbf{A} \rightarrow \mathbf{BC}$  the stack of  $\mathbf{A}$  is copied to both  $\mathbf{B}$  and  $\mathbf{C}$ . Thus, each rule is essentially a shorthand for an infinite family of derivation rules. The other two rules are used to pop and push onto the stack, respectively. We write variables in bold upper case, terminals in lower case, and indices by subscript. It is convenient to have a special symbol to mark the bottom of the index stack (corresponding to the rightmost subscript in an index string). We use  $\$$  for this purpose.

This class of languages properly includes all context-free grammars: these are the production rules with no indices. Furthermore, it is a proper subset of the class of context-sensitive languages (for instance  $\{(ab^n)^n : n \geq 0\}$  is context-sensitive but not indexed [10]).

As alluded to above, this is a full abstract family of languages which is closed under union, concatenation, Kleene closure, homomorphism, inverse homomorphism and intersection with regular sets. The set of indexed languages, however is not closed under intersection or complement.

---

*Date:* 24 October 2011.

*Key words and phrases.* Indexed grammars, generating functions, formal language theory. MM Gratefully acknowledges the support of NSERC Discovery Grant funding (Canada).

This class of languages is strictly larger than the class of context-free languages since it contains the language  $\{a^n b^n c^n : n > 0\}$ . A corresponding indexed grammar is given by

$$\mathbf{S} \rightarrow a\mathbf{A}_\$c \quad \mathbf{A} \rightarrow a\mathbf{A}_f c \quad \mathbf{A} \rightarrow \mathbf{B} \quad \mathbf{B}_\$ \rightarrow b \quad \mathbf{B}_f \rightarrow b\mathbf{B}.$$

We remark that for a typical derivation there is an initial pushing/loading stage to build up the stack, followed by a transfer stage; and finally a popping/unloading stage, to convert the indices into terminal symbols. The set of all languages generated by indexed grammars forms the set of indexed languages. The standard machine type that accepts the class of indexed languages is the nested stack automaton.

Formal language theory in general and indexed languages in particular have applications to group theory. Two good survey articles are [22] and [11]. Bridson and Gilman [2] have exhibited indexed grammar combings for fundamental 3-manifold groups based on Nil and Sol geometries (see Example 14 below). More recently [13] showed that the language of words in the standard generating set of the Grigorchuk group that do not represent the identity (the so-called *co-word* problem) forms an indexed language. The original DSV method (attributed to Delest, Schützenberger, and Viennot [5]) of computing the growth of a context-free language was successfully exploited [9] to compute the algebraic but non-rational growth series of a family of groups attributed to Higman. One of our goals is to extend this method to indexed grammars to deduce results on growth series.

**1.2. Ordinary generating functions.** Generating functions are a useful tool to treat enumerative questions about languages over finite alphabets, in particular the number of words of a given length. They permit a natural translation between combinatorial rules and functional equations. For any language  $\mathfrak{L}$ , let  $L_n$  be the number of words of length  $n$ . The ordinary generating function of the language is the formal power series  $L(z) = \sum_{n \geq 0} L_n z^n$ . We use this terminology interchangeably with growth series. This is a natural object to study in this context because of parallels of structure: regular languages have rational generating functions of the form  $L(z) = P(z)/Q(z)$  where  $P$  and  $Q$  are polynomials with integer coefficients. Unambiguous context-free languages have algebraic generating functions which satisfy  $P(L(z), z)$  for some bivariate polynomial  $P(x, y)$  with integer coefficients. The notion of complexity augments in parallel. One natural question to ask is the nature of the ordinary generating functions of languages derived from indexed grammars. Two natural candidates are D-finite, and differentiably algebraic. A formal series in  $\mathbb{C}[[z]]$  is said to be *D-finite* if it satisfies a homogeneous linear differential equation with polynomial coefficients. This is equivalent to a series with P-recursive coefficients. These series appear frequently as generating functions of structured combinatorial objects, although no direct interpretation is known. The class strictly contains the set of algebraic series. Most of our examples here are not D-finite because they contain an infinite number of singularities, a property inconsistent with D-finiteness. Indeed, many of our examples are lacunary series, which have a natural boundary at the unit circle. For more on the parallel, and on D-finite functions, consult [8, Appendix B.4]. A series  $L(z)$  is said to be *differentiably algebraic* if there is a  $k + 1$ -variate polynomial  $P(x_0, x_1, \dots, x_k)$

with integer coefficients such that  $P\left(L(z), \frac{d}{dz}L(z), \frac{d^2}{dz^2}L(z), \dots, \frac{d^k}{dz^k}L(z)\right) \equiv 0$ . Although this is a very wide class, at least one of our examples does not lie in this category.

**1.3. Summary.** Ideally, we would like to describe an efficient algorithm to determine the generating function of a given indexed grammar. We do describe a process in the next section that works under some conditions including only one stack symbol. Lemma 2 summarizes the conditions, and the results. We have several examples to illustrate the procedure. In Section 3 this is generalized to multiple stack symbols that are pushed in order. In this section we also illustrate the inherent obstacles in the case of multiple stack symbols. This is followed by some further examples from number theory in Section 4 and a discussion in Section 5 on inherent ambiguity in indexed grammars.

**1.4. Notation.** We use standard terminology with respect to formal language theory. The expression  $x|y$  denotes “ $x$  exclusive-or  $y$ ”. We use epsilon “ $\varepsilon$ ” to denote the empty word. The Kleene star operator applied to  $x$ , written  $x^*$  means make zero or more copies of  $x$ . A related notation is  $x^+$  which means make one or more copies of  $x$ . The word reversal of  $w$  is indicated by  $w^R$ . The length of the string  $x$  is denoted  $|x|$ . We print grammar variables in upper case bold letters. Grammar terminals are in lower case italic. We use the symbol  $\xrightarrow{*}$  to indicate the composition of two or more grammar productions.

## 2. THE METHOD: ONE INDEX SYMBOL

**2.1. Generalizing the DSV process.** Our starting point is [4]. Let  $\mathfrak{G}$  be a context-free grammar for some non-empty language  $\mathfrak{L}$ , with start symbol  $\mathbf{S}$ . Replace each terminal with the formal variable  $z$  and each non-terminal  $\mathbf{V}$  with a formal power series  $V(z)$ . Translate the grammar symbols  $\rightarrow, |, \varepsilon$  into  $=, +, 1$ , respectively, with juxtaposition becoming commutative multiplication. Thus the grammar is transformed into a system of equations. We summarize their main result as follows.

**Theorem.** *Each formal power series  $V(z) = \sum v_n z^n$  in the above transformation is an ordinary generating function where  $v_n$  is an integer representing the number of word productions of length  $n$  that can be realized from the non-terminal  $\mathbf{V}$ . In particular, if the original context-free grammar is unambiguous, then  $S(z)$  is the growth series for the language  $\mathfrak{L}$ , in which case  $S(z)$  is an algebraic function.*

A context-free grammar has only finitely many non-terminals. In the case of an indexed grammar, we treat a single variable  $\mathbf{V}$  as housing recursively many non-terminals, one for each distinct index string carried by  $\mathbf{V}$  (although only finitely many are displayed in parsing any given word). To generalize the DSV procedure to indexed grammars we apply the same transformation scheme to the grammar except that each variable with index string becomes a distinct function (for example  $\mathbf{V}_{gfgghgfs}$  becomes  $V_{gfgghgfs}(z)$ ).

The transformation recipe produces a system of infinitely many equations in infinitely many functions. It is not immediately apparent that such a system can be solved for  $S(z)$ , indeed we do not always obtain satisfying expressions. Let us illustrate a successful instance of the procedure with a simple example.

**Example 1.** The language  $\mathfrak{L}_{sq} = \{0^{2^n} : n \geq 0\}$  is generated by an indexed grammar. We use  $\$$  to indicate the bottom-most index symbol. Disregarding this, there is only one index symbol actually used:

$$\mathbf{S} \rightarrow \mathbf{T}_\$ \quad \mathbf{T} \rightarrow \mathbf{T}_f | \mathbf{D} \quad \mathbf{D}_f \rightarrow \mathbf{D}\mathbf{D} \quad \mathbf{D}_\$ \rightarrow 0$$

Observe that indices are loaded onto  $\mathbf{T}$  then transferred to  $\mathbf{D}$  which is then repeatedly doubled ( $\mathbf{D}$  is a mnemonic for “duplicator”). After all doubling, each instance of  $\mathbf{D}_\$$  becomes a zero.

The index strings used are particularly simple, consisting of a number of  $f$ ’s followed by one  $\$$ . This allows us to denote the functions  $V_{f^n\$}(z)$  by the simpler notation  $V_n(z)$  for  $V \in \{T, D\}$  and  $n \geq 1$ . The grammar rule  $\mathbf{D}_f \rightarrow \mathbf{D}\mathbf{D}$  implies the functional equality  $D_{n+1}(z) = D_n^2(z)$  and thus  $D_n(z) = D_0^{2^n}(z) = z^{2^n}$ .

The system of grammar equations become

$$S(z) = T_0(z) = T_1(z) + D_0(z) = T_2 + D_1 + D_0 = \cdots = \sum_{n \geq 0} D_n(z) = \sum_{n \geq 0} z^{2^n}$$

where we observe that the sequence of partial sums converge as a power series. We refer to this process, summarized in Lemma 2 below, as *pushing the  $T_n(z)$  off to infinity*. Note that  $S(z)$  is not an algebraic function. In fact,  $S(z)$  satisfies no algebraic differential equation [16].

**2.2. A straightforward case: Balanced indexed grammars.** Next, we describe a condition that allows us to guarantee that this process will result in a simplified generating function expression for  $S(z)$ .

Assume  $\mathbf{V}$  is a non-terminal in an indexed grammar  $\mathfrak{G}$ . We say  $\mathbf{V}$  *loads* or *pushes indices* if  $\mathfrak{G}$  contains a production having form  $\mathbf{V} \rightarrow \mathcal{U}\mathbf{V}_f\mathcal{U}'$  where  $\mathcal{U}$  and  $\mathcal{U}'$  each denote a (possibly empty) string of terminals and/or non-terminals and  $f$  is an index symbol. We say that  $f$  is the index symbol that is loaded or pushed onto  $\mathbf{V}$ . *Unloading* or *popping* indices means a production having form  $\mathbf{V}_f \rightarrow \mathcal{U}$ , where  $\mathcal{U}$  is a nonempty string of terminals and/or non-terminals. Call an indexed grammar *reduced* if at most one index symbol is loaded or unloaded for any given production and there are no useless non-terminals (every non-terminal  $\mathbf{V}$  satisfies both  $S \xrightarrow{*} \mathcal{U}\mathbf{V}_\sigma\mathcal{U}'$  and  $\mathbf{V}_\sigma \xrightarrow{*} w$  for some index string  $\sigma$  and some terminal string  $w$ ). A grammar is  $\varepsilon$ -free if the only production involving the empty string  $\varepsilon$  is  $\mathbf{S} \rightarrow \varepsilon$ . Indexed grammars  $\mathfrak{G}_1$  and  $\mathfrak{G}_2$  are *equivalent* if they produce the same language  $\mathfrak{L}$ .

**Theorem ([18]).** *Every indexed grammar  $\mathfrak{G}$  is equivalent to some reduced,  $\varepsilon$ -free grammar  $\mathfrak{G}'$ . Furthermore, there is an effective algorithm to convert  $\mathfrak{G}$  to  $\mathfrak{G}'$ .*

Consequently we will assume all grammars are already reduced (most of our examples are). We have found that  $\varepsilon$ -productions are a useful crutch in designing grammars (several of our examples are not  $\varepsilon$ -free). Our methods require an additional hypothesis.

**Definition.** An indexed grammar is *balanced* provided there are constants  $C, K \geq 0$ , depending only on the grammar, such that the longest string of indices associated to any non-terminal in any sentential form  $\mathcal{W}$  has length at most  $C|w| + K$  where  $w$  is any terminal word produced from  $\mathcal{W}$ . (Note: in all our balanced examples we can take  $C = 1$  and  $K \in \{0, 1\}$ .)

**Lemma 2.** *Let  $\mathfrak{G}$  be an unambiguous balanced indexed grammar for some non-empty language  $\mathfrak{L}$  that involves only one index symbol (say  $f$ ) and suppose  $\mathbf{V}$  is the only non-terminal that loads  $f$ . Then in the generalized DSV equations for  $\mathfrak{G}$ , the sequence of functions  $V_n(z)$  can be eliminated (pushed to infinity), where  $n > 0$  refers to the number of  $f$  indices on  $V$ . Under these hypotheses, the system of equations defining  $S(z)$  reduces to finitely many bounded recurrences with initial conditions whose solution is the growth function for  $\mathfrak{L}$ .*

*Proof.* Consider all productions in  $\mathfrak{G}$  that have  $\mathbf{V}$  on the left side. Converting these productions into the usual functional equations gives an equation of form either

$$V_n(z) = V_{n+1}(z) + W_{n\pm e}(z) \quad \text{or} \quad V_n(z) = V_{n+1}(z)W_{n\pm e}(z)$$

(here  $W_{n\pm e}(z)$  denotes an expression that represents all other grammar productions having  $\mathbf{V}$  on the left side while  $e \in \{0, 1, -1\}$ ). Consider the equation  $V_n(z) = V_{n+1}(z) + W_{n\pm e}(z)$ . Fix  $e = 0$  for the remainder of this paragraph. Starting with  $n = 0$  and iterating  $N \gg 0$  times yields  $V_0(z) = V_N(z) + W_0(z) + W_1(z) + \dots + W_N(z)$ . By the balanced hypothesis, there exists a constants  $C, K \geq 0$  such that all terminal words produced from  $\mathbf{V}_{fN}$  have length at least  $N/C - K \gg 0$ . This means that the first  $N/C - K$  terms in the ordinary generating function for  $V_0(z)$  are unaffected by the contributions from  $V_N(z)$  and depend only on the fixed sum  $W_0(z) + W_1(z) + \dots + W_N(z)$ . Therefore the  $(N/C - K)^{th}$  partial sum defining the generating function for  $V_0(z)$  is stabilized as soon as the iteration above reaches  $V_N(z)$ . This is true for all big  $N$ , so we may take the limit as  $N \rightarrow \infty$  and express  $V_0(z) = \sum_{n \geq 0} W_n(z)$ .

If the equation for loading of index  $f$  takes form  $V_n(z) = V_{n+1}(z)W_n(z)$ , a similar argument derives  $V_0(z) = \prod_{n \geq 0} W_n(z)$ . Allowing  $e = \pm 1$  in either form merely shifts indices in the sum/product and does not affect the logic of the argument. Therefore, in all cases the variables  $V_n(z)$  are eliminated from the system of equations induced by the grammar  $\mathfrak{G}$ . We assumed that  $\mathbf{V}$  was the only variable loading indices, so all other grammar variables either unload/pop indices (yielding finitely many finite recurrences of bounded depth) or evaluate as terminals (supplying recurrence initial conditions). The solution of this simplified system is  $S(z)$ .  $\square$

It turns out that the balanced hypothesis used above is already satisfied.

**Lemma 3.** *Suppose the indexed grammar  $\mathfrak{G}$  is unambiguous, reduced,  $\varepsilon$ -free, and has only one index symbol  $f$  (other than  $\$$ ). Then  $\mathfrak{G}$  is balanced.*

*Proof.* If the language produced by  $\mathfrak{G}$  is finite, there is nothing to prove so we may assume the language is infinite. Let us define a special sequence of grammar productions used in producing a terminal word. Suppose a sentential form contains several variables, each of which is ready for unloading of the index symbol  $f$ . A *step* will consist of unloading a single  $f$  from each of these non-terminals, starting from the leftmost variable. After the step, each of these variables will hold an index string that is exactly one character shorter than before.

Consider a sentential form  $F$  containing one or more non-terminals in the unloading stage and each of whose index strings are of length at least  $N \gg 0$ . These symbols can only be unloaded at the rate of one per production rule (this is the reduced hypothesis) and we'll only consider changing  $F$  by steps.

On the other hand there, are only finitely many production rules to do this unloading of finitely many variables. Thus for large  $N$  there is a cycle of production rules as indices are unloaded, with cycle length bounded by a global constant  $C > 0$  which depends only on the grammar. Furthermore, this cycle is reached after at most  $K < C$  many productions. Let  $F'$  denote the sentential form that results from  $F$  as one such cycle is begun and let  $F''$  be the sentential form after the cycle is applied to  $F'$ .

Consider lengths of sentential forms (counting terminals and variables but ignoring indices). Since the grammar is reduced and  $\varepsilon$ -free, each grammar production is a non-decreasing function of lengths. Thus  $|F''| \geq |F'| \geq |F|$ . Discounting any indices, the equality of sentential forms  $F'' = F'$  is not possible because this implies ambiguity.

We claim that either  $F''$  is longer than  $F'$  or that  $F''$  has more terminals than  $F'$ . If not, then  $F''$  has exactly the same terminals as  $F'$ , and each has the same quantity of variables. There are only finitely many arrangements of terminals and variables for this length and for large  $N$  we may loop stepwise through the production cycle arbitrarily often and thus repeat exactly a sentential form (discounting indices). This implies our grammar is ambiguous contrary to hypothesis.

Thus after  $C$  steps the sentential forms either obtain at least one new terminal or non-terminal. In the latter case, variables must convert into terminals on  $\$$  (via the reduced, unambiguous,  $\varepsilon$ -free hypotheses). There will be at least one terminal per step in the final output word  $w$ . We obtain the inequality  $(N - K)/C \leq |w|$  which establishes the lemma.  $\square$

**2.3. A collection of examples.** We illustrate the method, its power and its limitations with three examples: the first two are examples with intermediate growth, and one which we are unable to resolve to our satisfaction.

The first example is originally due to [12] and features the indexed language  $\mathcal{L}_{G/M} = \{ab^{i_1}ab^{i_2}\dots ab^{i_k} : 0 \leq i_1 \leq i_2 \leq \dots \leq i_k\}$  with intermediate growth (meaning that the number of words of length  $n$  ultimately grows faster than any polynomial in  $n$  but more slowly than  $2^{kn}$  for any constant  $k > 0$ ). The question of whether a context-free language could have this property was asked in [7] and answered in the negative [15, 3]. Grigorchuk and Machí constructed their language based on the generating function of Euler's partition function. A word of length  $n$  encodes a partition sum of  $n$ . For instance, the partitions of  $n = 5$  are  $1+1+1+1+1$ ,  $1+1+1+2$ ,  $1+2+2$ ,  $1+1+3$ ,  $2+3$ ,  $1+4$ ,  $5$ . The corresponding words in  $\mathcal{L}_{G/M}$  are  $aaaaa$ ,  $aaaab$ ,  $aabab$ ,  $aaabbb$ ,  $ababb$ ,  $aabbb$ ,  $abbbb$ , respectively. The derivation below is ours.

**Example 4.** An unambiguous grammar for  $\mathcal{L}_{G/M}$  is

$$\mathbf{S} \rightarrow \mathbf{T}_\$ \quad \mathbf{T} \rightarrow \mathbf{T}_f|\mathbf{GT}|\mathbf{G} \quad \mathbf{G}_f \rightarrow \mathbf{Gb} \quad \mathbf{G}_\$ \rightarrow a$$

The latter two productions imply that  $\mathbf{G}_{f\$} \xrightarrow{*} ab^m$  or in terms of functions  $G_m(z) = z^{m+1}$ . A typical parse tree is illustrated in Figure 2.1.

The second grammar production group transforms to

$$T_m(z) = T_{m+1}(z) + G_m(z)T_m(z) + G_m(z) .$$

Substitution and solving for  $T_m$  gives

$$T_m(z) = \frac{z^{m+1} + T_{m+1}(z)}{1 - z^{m+1}} .$$

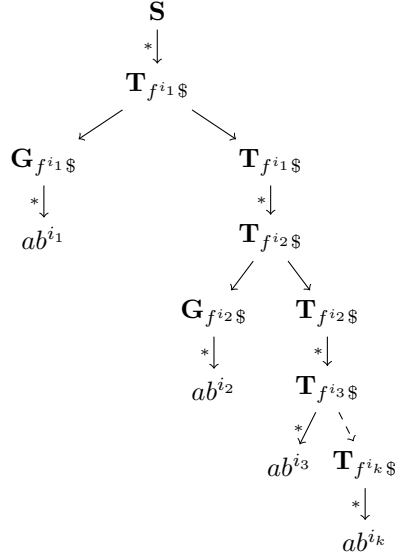


FIGURE 2.1. A typical parse tree in the grammar

$$\mathbf{S} \rightarrow \mathbf{T}_{\$} \quad \mathbf{T} \rightarrow \mathbf{T}_f | \mathbf{GT} | \mathbf{G} \quad \mathbf{G}_f \rightarrow \mathbf{G}b \quad \mathbf{G}_{\$} \rightarrow a$$

Iterating this recurrence yields a kind of inverted continued fraction:

$$S(z) = T_0(z) = \frac{z + T_1(z)}{1 - z} = \frac{z + \frac{z^2 + T_2(z)}{1 - z^2}}{1 - z} = \frac{z + \frac{z^2 + \frac{z^3 + \dots}{1 - z^3}}{1 - z^2}}{1 - z}.$$

Equivalently, this recurrence can be represented as

$$\frac{z + T_1(z)}{1 - z} = \frac{z(1 - z^2) + z^2 + T_2(z)}{(1 - z)(1 - z^2)} = \frac{z(1 - z^2)(1 - z^3) + z^2(1 - z^3) + z^3 + T_3(z)}{(1 - z)(1 - z^2)(1 - z^3)}$$

or

$$S(z) = \frac{z}{1 - z} + \frac{z^2}{(1 - z)(1 - z^2)} + \frac{z^3}{(1 - z)(1 - z^2)(1 - z^3)} + \dots + \frac{z^k + T_k(z)}{\prod_{n=1}^k (1 - z^n)}.$$

Apply Lemma 2 to push  $T_k(z)$  off to infinity:

$$S(z) = \sum_{j \geq 1} \frac{z^j}{(1 - z)(1 - z^2) \dots (1 - z^j)}.$$

Here we recognize a classic combinatorial summation of partitions in term of their largest part [8, Example I.7]. Thus, we have recovered the ordinary generating function for partitions,  $S(z) = \sum_{n \geq 1} p(n)z^n$ , where the coefficients belong to Euler's partition sequence  $p(n)$ . Since we can also write

$$\sum_{n \geq 1} p(n)z^n = \prod_{n \geq 1} \frac{1}{1 - z^n},$$

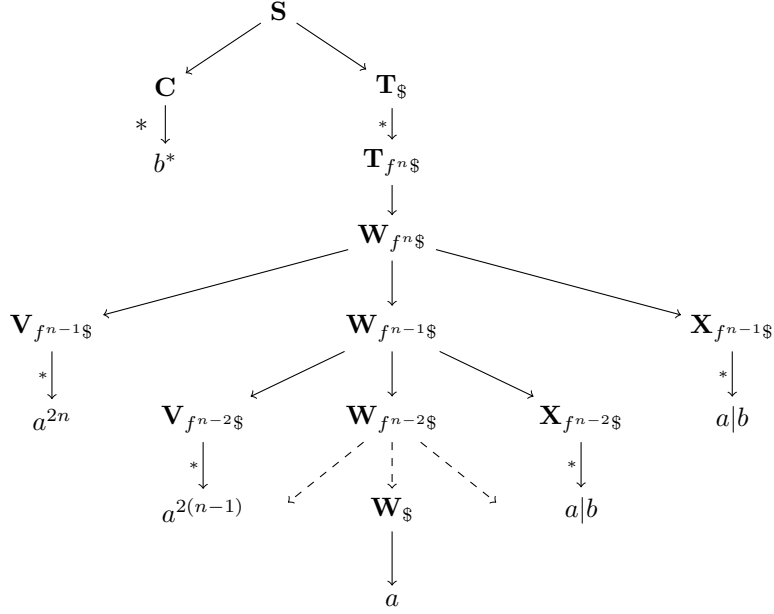


FIGURE 2.2. A typical parse tree in the grammar

$$\begin{aligned}
 S &\rightarrow C|CT_{\$} & C &\rightarrow bC|\varepsilon & T &\rightarrow T_f|W & W_f &\rightarrow VWX \\
 V_f &\rightarrow aaV & V_{\$} &\rightarrow aa & W_{\$} &\rightarrow a & X &\rightarrow a|b
 \end{aligned}$$

it is true that  $S(z)$  has a dense set of singularities on the unit circle and is not D-finite.

**Example 5.** Another series with intermediate growth can be realized as the ordinary generating function of the following indexed grammar:

$$\begin{aligned}
 S &\rightarrow C|CT_{\$} & C &\rightarrow bC|\varepsilon & T &\rightarrow T_f|W & W_f &\rightarrow VWX \\
 V_f &\rightarrow aaV & V_{\$} &\rightarrow aa & W_{\$} &\rightarrow a & X &\rightarrow a|b
 \end{aligned}$$

As usual, we use index  $\$$  to indicate the bottom of the stack, with  $f$  being the only actual index symbol. A typical parse tree is given in Figure 2.2. From this we see that the language generated (unambiguously) is

$$\mathcal{L}_{int} = \left\{ b^* \left( \varepsilon \mid a^{n^2+n+1} (a|b)^n \right) : n \geq 0 \right\}.$$

We can derive the generating function in the usual fashion. Note the shortcuts  $X_{f^n\$} \rightarrow (a|b)$  (regardless of indices) and  $V_{f^n\$} \xrightarrow{*} a^{2n} V_{\$} \rightarrow a^{2n+2}$ . Starting with

$$W_{ff\$} \rightarrow V_f\$ W_{f\$} X_{f\$} \xrightarrow{*} a^4 W_{f\$} (a|b) \rightarrow a^4 V_{\$} W_{\$} X_{\$} (a|b) \xrightarrow{*} a^4 a^2 a (a|b)^2$$

one can use induction to derive

$$W_{f^n\$} \xrightarrow{*} a^{2n} \cdots a^4 a^2 a (a|b)^n = a^{n(n+1)} a (a|b)^n = a^{n^2+n+1} (a|b)^n.$$



In terms of generating functions these shortcuts imply  $W_n(z) = z^{n^2+n+1}2^n z^n = 2^n z^{(n+1)^2}$ ; also  $C(z) = \frac{1}{1-z}$ . Put this all together to get

$$\begin{aligned} S(z) &= C(z) + C(z)T_0(z) = C(1 + T_0) = C(1 + T_1 + W_0) = \\ &= C(1 + T_2 + W_0 + W_1) = \cdots = C\left(1 + \sum_{n=0}^{\infty} W_n\right) \\ &= \frac{1}{1-z} \left(1 + \sum_{n=0}^{\infty} 2^n z^{(n+1)^2}\right). \end{aligned}$$

Write as a sum of rational functions and expand each geometric series:

$$\begin{aligned} S(z) &= \frac{1}{1-z} + \frac{z}{1-z} + \frac{2z^4}{1-z} + \frac{4z^9}{1-z} + \frac{8z^{16}}{1-z} + \cdots \\ &= \begin{array}{ccccccc} 1+z+z^2+z^3+z^4 & +z^5 & +z^6 & +z^7 & +z^8 & +z^9 & +\cdots \\ +z+z^2+z^3+z^4 & +z^5 & +z^6 & +z^7 & +z^8 & +z^9 & +\cdots \\ & +2z^4 & +2z^5 & +2z^6 & +2z^7 & +2z^8 & +2z^9 & +\cdots \\ & & & & & +4z^9 & +\cdots \\ & & & & & & \ddots \end{array} \end{aligned}$$

and so forth. Sum the columns and observe that the coefficient of each  $z^n$  is a power of 2, with new increments occurring when  $n$  is a perfect square. Thus

$$S(z) = \sum_{n=0}^{\infty} 2^{\lfloor \sqrt{n} \rfloor} z^n$$

and the coefficient of  $z^n$  grows faster than any polynomial (as  $n \rightarrow \infty$ ) but is sub-exponential.

The indexed grammars used in applications (combings of groups, combinatorial descriptions, etc) tend to be reasonably simple and most use one index symbol. Despite the success of our many examples, Lemma 2 does *not* guarantee that an explicit closed formula for  $S(z)$  can always be found.

**Example 6.** Consider the balanced grammar below.

$$\mathbf{S} \rightarrow \mathbf{T}_{\S} \quad \mathbf{T} \rightarrow \mathbf{T}_f | \mathbf{N} \quad \mathbf{N}_f \rightarrow a\mathbf{N} | b^2\mathbf{M} \quad \mathbf{M}_f \rightarrow ab\mathbf{NM} \quad \mathbf{M}_{\S}, \mathbf{N}_{\S} \rightarrow \varepsilon$$

The hypotheses of Lemma 2 are satisfied so we can push  $T$  to infinity and obtain

$$S(z) = T_0 = N_0 + T_1 = N_0 + N_1 + T_2 = \cdots = \sum_{n \geq 0} N_n(z).$$

However, the recursions defining  $N_n(z)$  are intertwined and formidable:

$$N_n(z) = zN_{n-1} + z^2M_{n-1} \quad \text{and} \quad M_n(z) = z^2N_{n-1}M_{n-1} \quad \forall n \geq 1$$

with  $N_0 = 1 = M_0$ . It is possible to eliminate  $M$  but the resulting nonlinear recursion

$$N_n(z) = zN_{n-1} + z^2N_{n-1}N_{n-2} - z^3N_{n-2}^2$$

does not appear to be a bargain (it is possible that a multivariate generating function as per Example 14 may be helpful). The reader is invited to invent grammars of this type with more difficult recursions.

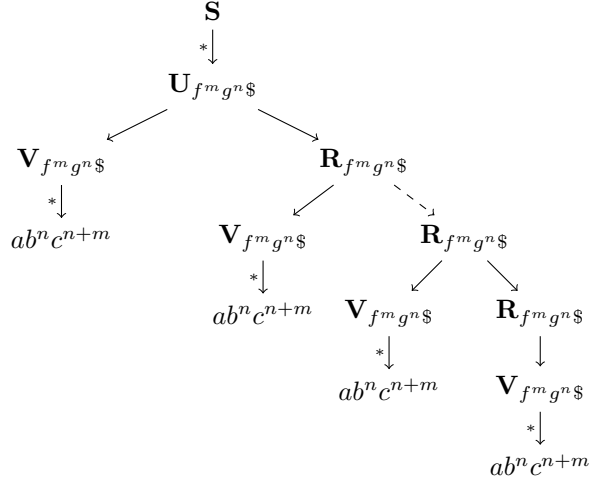


FIGURE 3.1. A typical parse tree in the grammar

$$\begin{aligned}
S &\rightarrow T_{\$} & T &\rightarrow T_g | U_f & U &\rightarrow U_f | VR | V & R &\rightarrow VR | V \\
V &\rightarrow aBC & B_f &\rightarrow Bb & B_g &\rightarrow \varepsilon & C_f &\rightarrow Cc & C_g &\rightarrow cC & C_{\$} &\rightarrow \varepsilon
\end{aligned}$$

### 3. THE CASE OF SEVERAL INDEX SYMBOLS

Our next example uses two index symbols (in addition to  $\$$ ) in an essential way.

**Example 7.** Define  $\mathcal{L}_{serial} = \{(ab^i c^j)^+ : 1 \leq i \leq j\}$ . Consider the grammar:

$$\begin{aligned}
S &\rightarrow T_{\$} & T &\rightarrow T_g | U_f & U &\rightarrow U_f | VR | V & R &\rightarrow VR | V \\
V &\rightarrow aBC & B_f &\rightarrow Bb & B_g &\rightarrow \varepsilon & C_f &\rightarrow Cc & C_g &\rightarrow cC & C_{\$} &\rightarrow \varepsilon
\end{aligned}$$

Observe that the two index symbols are loaded serially: all  $g$ 's are loaded prior to any  $f$  so each valid index string will be of the form  $f^+ g^* \$$ . We also have the shortcuts  $C_{f^m g^n \$} \xrightarrow{*} c^m C_{g^n \$} \xrightarrow{*} c^{m+n}$  and  $B_{f^m g^n \$} \xrightarrow{*} b^m B_{g^n \$} \rightarrow b^m \varepsilon = b^m$  and consequently  $V_{f^m g^n \$} \rightarrow a B_{f^m g^n \$} C_{f^m g^n \$} \xrightarrow{*} ab^m c^{m+n}$ . A typical parse tree is given in Figure 3.1.

The special form of the index strings ensures that such a string is uniquely identified solely by the number of  $f$ 's and number of  $g$ 's it carries. Consequently, the induced function  $V_{f^m g^n \$}(z)$  can be relabeled more simply as  $V_{m,n}(z)$ , and similarly with functions  $T, U, R$ . Working in the reverse order of the listed grammar productions, we have the identities

$$V_{m,n}(z) = z^{2m+n+1} \quad \text{and} \quad R_{m,n} = \frac{V_{m,n}}{1 - V_{m,n}} = \frac{z^{2m+n+1}}{1 - z^{2m+n+1}}.$$

The grammar production  $U \rightarrow U_f | VR | V$  implies for fixed  $n > 0$  that

$$U_{1,n}(z) = U_{2,n} + (V_{1,n} R_{1,n} + V_{1,n}) = U_{3,n} + (V_{2,n} R_{2,n} + V_{2,n}) + (V_{1,n} R_{1,n} + V_{1,n}).$$

The hypothesis of Lemma 2 are satisfied in that we are dealing with a balanced grammar where currently only one index symbol is being loaded onto one variable. Therefore for fixed  $n$  we can push  $U_{m,n}(z)$  off to infinity and obtain

$$U_{1,n}(z) = \sum_{m \geq 1} (V_{m,n} R_{m,n} + V_{m,n}) = \sum_{m \geq 1} R_{m,n}(z) = \sum_{m \geq 1} \frac{z^{2m+n+1}}{1 - z^{2m+n+1}}.$$

Our general derivation proceeds as follows:

$$S(z) = T_{0,0} = T_{0,1} + U_{1,0} = T_{0,2} + U_{1,1} + U_{1,0} = \cdots = T_{0,k+1} + \sum_{n=1}^k U_{1,n}$$

Lemma 2 can be invoked again to eliminate  $T$ . We find that

$$S(z) = \sum_{n \geq 1} \sum_{m \geq 1} \frac{z^{2m+n+1}}{1 - z^{2m+n+1}} = \sum_{j \geq 1} \frac{z^{3j}}{(1 - z^j)(1 - z^{2j})} = \sum_{i \geq 1} \frac{z^{3i} + z^{4i}}{(1 - z^{2i})^2}$$

with the latter two summations realized by expanding geometric series and/or changing the order of summation in the double sum. In any event,  $S(z)$  has infinitely many singularities on the unit circle and is not D-finite.

It is worth noting the reason why the copying schema used above succeeds in indexed grammars (but fails for context-free grammars). The word  $ab^i c^j$  is first encoded as an index string attached to  $\mathbf{V}$  and only then copied to  $\mathbf{VR}$  (the grammar symbol  $\mathbf{R}$  is a mnemonic for “replicator”). This ensures that  $ab^i c^j$  is faithfully copied. Slogan: “encode first, then copy”. Context-free grammars are limited to “copy first, then express” which does not allow for fidelity in copying.

We would like to generalize the previous example. The key notion was the manner in which the indices were loaded.

**Definition.** Suppose that  $\mathfrak{G}$  is an unambiguous indexed grammar with index alphabet  $I = \{f_1, f_2, \dots, f_n\}$  such that every index string  $\sigma$  has form  $f_n^* f_{n-1}^* \cdots f_1^* \$$ . We say the indices are *loaded serially* in  $\mathfrak{G}$ .

**Corollary 8.** Assume  $\mathfrak{G}$  is an unambiguous balanced indexed grammar with variable set  $V$ , non-terminal alphabet  $\Sigma$ , and index alphabet  $I = \{f_1, \dots, f_n\}$ . Suppose all indices are loaded serially onto respective variables  $\mathbf{T}^{\{1\}}, \dots, \mathbf{T}^{\{n\}}$  and in the indicated order. Then each function family  $T^{\{j\}}(z)$  can be eliminated (“pushed to infinity”) and the system of equations defining  $S(z)$  can be reduced to finitely many recursions as per the conclusion of Lemma 2.

*Proof.* We have assumed that  $\mathbf{T}^{\{n\}}$  is the last variable to load indices and is loaded with  $f_n$  only, so  $\mathbf{T}^{\{n\}}$  carries an index string  $\sigma$  of form  $f_n^* \cdots f_2^* f_1^* \$$ . Indeed, any grammar production having  $\mathbf{T}^{\{n\}}$  on the left side will have a right side of two types: a string  $\mathcal{U}1 \in (V|\Sigma)^*$  that loads  $f_n$  onto  $\mathbf{T}^{\{n\}}$  or a string  $\mathcal{U}2 \in (V|\Sigma)^*$  without any loading of indices. Neither of these types will include any variable  $\mathbf{T}^{\{j\}}$  for  $j < n$  (by the serial loading hypothesis) nor will there be any unloading of indices (by the unambiguous hypothesis). Consequently, the equations having  $T_k^{\{n\}}(z)$  on the left side have on their right side products and sums involving no  $T^{\{j\}}(z)$  for  $j < n$  but only  $T_k^{\{n\}}(z)$  and functions that define finite recurrences. The hypotheses of Lemma 2 apply to this situation and  $T^{\{n\}}$  can be pushed to infinity.

The previous paragraph is both the basis step and induction step of an obvious argument that eliminates  $T^{\{n-1\}}$  then  $T^{\{n-2\}}$  and so on till  $T^{\{1\}}$ .  $\square$

In the case of a grammar with multiple index symbols, we would like to be able to replace an unwieldy expression like  $A_{ghghgf\$}(z)$  with  $A_{2,3,1}(z)$  where the subscripts indicate two occurrences of  $f$ , three of  $g$ , and one  $h$ . This is certainly possible for a grammar with only one index symbol (excluding the end of stack marker  $\$$ ) or several symbols loaded serially as per the previous Corollary, but is not possible in general.

**Example 9.** (Ordering matters) Consider the language  $\mathcal{L}_{ord}$  generated by the indexed grammar below.

$$\mathbf{S} \rightarrow \mathbf{T}_\$ \quad \mathbf{T} \rightarrow \mathbf{T}_\alpha | \mathbf{T}_\beta | \mathbf{N} \quad \mathbf{N}_\alpha \rightarrow a\mathbf{N} \quad \mathbf{N}_\beta \rightarrow b\mathbf{N}b\mathbf{N}b \quad \mathbf{N}_\$ \rightarrow \varepsilon$$

When applying the DSV transformations to this grammar we would like to write  $N_{1,1}(z)$  as the formal power series corresponding to the grammar variable  $\mathbf{N}$  with any index string having one  $\alpha$  index and one  $\beta$  index, followed by the end of stack marker  $\$$ . Note the derivations  $\mathbf{S} \xrightarrow{*} \mathbf{N}_{\alpha\beta\$} \xrightarrow{*} abbb$  and  $\mathbf{S} \xrightarrow{*} \mathbf{N}_{\beta\alpha\$} \xrightarrow{*} babab$ . Even though both intermediate sentential forms have one of each index symbol, followed by the end of stack marker  $\$$ , they produce distinct words of differing length. Thus using subscripts to indicate the quantity of stack indices cannot work in general without some consideration of index ordering.

We note that the grammar is reduced and balanced. It is also unambiguous, which can be verified by induction. In fact, if  $\sigma \in (\alpha|\beta)^*\$$  is an index string such that  $\mathbf{N}_\sigma \xrightarrow{*} w$  where  $w$  is a terminal word of length  $n$ , then  $\mathbf{N}_{\alpha\sigma} \xrightarrow{*} aw$  and  $\mathbf{N}_{\beta\sigma} \xrightarrow{*} bwbw$  where  $|aw| = n + 1$  and  $|bwbw| = 2n + 3$ . Suppose that all words  $w \in \mathcal{L}_{ord}$  of length  $n$  or less are produced unambiguously. Consider a word  $v$  of length  $n + 1$ . Either  $v = aw$  or  $v = bw'bw'b$  for some shorter words  $w, w' \in \mathcal{L}_{ord}$  that were produced unambiguously by hypothesis. Clearly neither of these forms for  $v$  can be confused since one starts with  $a$  and the other with  $b$ .

The proof of Lemma 2 can be applied to eliminate the  $T_\sigma(z)$ . Solving for  $S(z)$  via the generalized DSV procedure gives

$$S(z) = \sum_{\sigma \in I} N_\sigma(z)$$

where the sum is over all index strings  $\sigma$ . It is unfeasible to simplify further because the number of grammar functions  $N_\sigma(z)$  grows exponentially in the length of  $\sigma$  without suitable simplifying recursions.

The previous example showed two non-terminals having index strings with the same quantity of respective symbols but in different orders leading to two distinct functions. We can define a condition that ensures such functions are the same.

**Definition.** Let  $\mathcal{A} = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$  denote a finite alphabet. The *Parikh vector* associated to  $\sigma \in \mathcal{A}$  records the number of occurrences of each  $\alpha_i$  in  $\sigma$  as  $x_i$  in the vector  $[x_1, x_2, \dots, x_n]$  (see [19]). Define two strings  $\sigma, \tau \in \mathcal{A}$  to be *Parikh equivalent* if they map to the same Parikh vector (in other words  $\tau$  is a permutation of  $\sigma$ ). When  $\mathcal{A}$  is the index alphabet for a grammar, we extend this idea to non-terminals and say  $\mathbf{V}_\sigma$  is Parikh equivalent to  $\mathbf{V}_\tau$  if  $\sigma$  and  $\tau$  map to the same Parikh vector.

The following lemma gives sufficient conditions that allow simplifying the ordering difficulty for function subscripts. Its proof is immediate.

**Lemma 10.** *Assume  $\mathfrak{G}$  is a nontrivial balanced indexed grammar. Suppose each pair of Parikh equivalent index strings  $\sigma, \tau$  appended to a given grammar variable  $\mathbf{V}$  result in identical induced functions  $V_\sigma(z) \equiv V_\tau(z)$ . Then the functions induced from  $\mathbf{V}$  can be consolidated into equivalence classes (where we replace index string subscripts by their respective Parikh vectors) without changing the solution  $S(z)$  of the system of DSV equations.*

We have already used this lemma in Example 7 above. We illustrate with another example.

**Example 11.** Consider the non-context-free language  $\mathcal{L}_{double} = \{ww : w \in (a|b)^*\}$  produced by the grammar

$$\mathbf{S} \rightarrow \mathbf{T}_\$ \quad \mathbf{T} \rightarrow \mathbf{T}_\alpha | \mathbf{T}_\beta | \mathbf{R}\mathbf{R} \quad \mathbf{R}_\alpha \rightarrow a\mathbf{R} \quad \mathbf{R}_\beta \rightarrow b\mathbf{R} \quad \mathbf{R}_\$ \rightarrow \varepsilon .$$

Suppose  $\sigma, \tau \in (\alpha|\beta)^*\$$  are Parikh equivalent index strings of length  $n$ . It is clear that  $R_\sigma(z) = z^n = R_\tau(z)$ . In fact, every string  $u \in (\alpha|\beta)^n\$$  implies  $R_u(z) = z^n$ , regardless of the particular distribution of  $\alpha$  and  $\beta$  in  $u$ . Instead of using the equivalence classes  $R_{i,j}(z)$  where  $[i, j]$  is the Parikh vector for  $u$ , let  $R_n(z)$  denote the equivalence class of all such induced functions  $R_u(z)$  where  $u \in (\alpha|\beta)^n\$$ , and define  $T_n(z)$  similarly. We will abuse notation and refer to the elements of these classes as  $R_n(z)$  or  $T_n(z)$ , respectively. The grammar equations become

$$S(z) = T_0 = 2T_1 + R_0^2 = R_0^2 + 2(2T_2 + R_1^2) = R_0^2 + 2R_1^2 + 4R_2^2 + \dots$$

where we can push the  $T_n(z)$  to infinity as per the proof of Lemma 2. Therefore

$$S(z) = \sum_{n \geq 0} 2^n R_n^2(z) = \sum_{n \geq 0} 2^n z^{2n} = \frac{1}{1 - 2z^2} .$$

#### 4. FURTHER EXAMPLES RELATED TO NUMBER THEORY

In addition to our example from [12] we have the following

**Example 12.** Define  $\mathcal{L}_{div} = \{a^n (b^n)^* : n > 0\}$  which is generated by the unambiguous balanced grammar

$$\begin{aligned} \mathbf{S} &\rightarrow \mathbf{T}_\$ & \mathbf{T} &\rightarrow \mathbf{T}_f | \mathbf{A}_f \mathbf{R}_f & \mathbf{R} &\rightarrow \mathbf{B}\mathbf{R} | \varepsilon \\ \mathbf{A}_f &\rightarrow a\mathbf{A} & \mathbf{A}_\$ &\rightarrow \varepsilon & \mathbf{B}_f &\rightarrow b\mathbf{B} & \mathbf{B}_\$ &\rightarrow \varepsilon \end{aligned}$$

We see some familiar shortcuts:  $\mathbf{A}_f n \$ \rightarrow a^n$  and  $\mathbf{B}_f n \$ \rightarrow b^n$ . In terms of functions this means  $A_n(z) = z^n = B_n(z)$  and furthermore  $R_n = B_n R_n + 1$  implies  $R_n(z) = \frac{1}{1 - z^n}$ . Thus our main derivation becomes

$$S(z) = T_0 = T_1 + A_1 R_1 = T_2 + A_1 R_1 + A_2 R_2 = \dots = \sum_{n \geq 1} A_n R_n = \sum_{n \geq 1} \frac{z^n}{1 - z^n} .$$

Expand each rational summand into a geometric series and collect terms

$$\begin{aligned}
S(z) &= \frac{z}{1-z} + \frac{z^2}{1-z^2} + \frac{z^3}{1-z^3} + \frac{z^4}{1-z^4} + \dots \\
&= \begin{array}{ccccccc}
z & +z^2 & +z^3 & +z^4 & +z^5 & +z^6 & +z^7+z^8+z^9+z^{10}+\dots \\
& +z^2 & & +z^4 & & +z^6 & +z^8 & +z^{10}+\dots \\
& & +z^3 & & +z^6 & & +z^9 & +\dots \\
& & & +z^4 & & +z^8 & & +\dots \\
& & & & +z^5 & & +z^{10}+\dots
\end{array} \\
&\quad \ddots \\
&= z + 2z^2 + 2z^3 + 3z^4 + 2z^5 + \dots
\end{aligned}$$

We see the table houses a sieve of Eratosthenes and we find that

$$S(z) = z + 2z^2 + 2z^3 + 3z^4 + 2z^5 + 4z^6 + \dots = \sum_{n \geq 1} \tau(n) z^n$$

where  $\tau(n)$  is the number of positive divisors of  $n$ . Again,  $S(z)$  has infinitely many singularities on the unit circle and is not D-finite.

**Example 13.** Let  $\mathfrak{L}_{comp} = \{0^c : c \text{ is composite}\}$  denote the composite numbers written in unary. A generative grammar is

$$\begin{aligned}
\mathbf{S} &\rightarrow \mathbf{T}_f \mathfrak{s} & \mathbf{T} &\rightarrow \mathbf{T}_f | \mathbf{R} & \mathbf{R} &\rightarrow \mathbf{R} \mathbf{A} | \mathbf{A} \mathbf{A} \\
\mathbf{A}_f &\rightarrow 0 \mathbf{A} & \mathbf{A}_{\mathfrak{s}} &\rightarrow 0
\end{aligned}$$

with sample derivation

$$\mathbf{S} \xrightarrow{*} \mathbf{R}_{f^n \mathfrak{s}} \rightarrow \mathbf{R}_{f^n \mathfrak{s}} \mathbf{A}_{f^n \mathfrak{s}} \xrightarrow{*} \mathbf{R}_{f^n \mathfrak{s}} (\mathbf{A}_{f^n \mathfrak{s}})^m \rightarrow (\mathbf{A}_{f^n \mathfrak{s}})^{m+1} \xrightarrow{*} 0^{(n+1)(m+1)}.$$

This is certainly an ambiguous grammar because there is a distinct production of  $0^c$  for each nontrivial factorization of  $c$ . (Note: one can tweak the grammar to allow the trivial factorizations  $1 \cdot c$  and  $c \cdot 1$ . The resultant language becomes the semigroup  $0^+$  isomorphic to  $\mathbb{Z}^+$  but the generating function of all grammar productions is the familiar  $\sum \tau(n) z^n$  which we saw in Example 12.)

Suppose we want the generating function for the *sum* of positive divisors  $\sum \sigma(n) z^n$ ? Then our table expansion above would look like

$$\begin{aligned}
S(z) &= \begin{array}{ccccccc}
z & +2z^2 & +3z^3 & +4z^4 & +5z^5 & +6z^6 & +\dots \\
& +2z^2 & & +4z^4 & & +6z^6 & +\dots \\
& & +3z^3 & & +6z^6 & +\dots \\
& & & +4z^4 & & +\dots
\end{array} \\
&\quad \ddots \\
&= z + 4z^2 + 6z^3 + 12z^4 + \dots
\end{aligned}$$

and now each row has closed form  $\frac{z^n}{(1-z^n)^2}$ . Our goal is to modify the grammar of the previous example to obtain  $\sum \frac{z^n}{(1-z^n)^2}$ . At first glance it would seem that we only need replace the grammar rule  $\mathbf{T} \rightarrow \mathbf{T}_f | \mathbf{A}_f \mathbf{R}_f$  with  $\mathbf{T} \rightarrow \mathbf{T}_f | \mathbf{A}_f \mathbf{R}_f \mathbf{R}_f$  which replaces  $S(z) = \sum A_n R_n$  with  $\sum A_n R_n R_n$ . However this creates ambiguity because we can produce  $ab$  in two ways:  $S \xrightarrow{*} A_f \mathfrak{s} R_f \mathfrak{s} R_f \mathfrak{s} \xrightarrow{*} ab\varepsilon$  and  $S \xrightarrow{*} A_f \mathfrak{s} R_f \mathfrak{s} R_f \mathfrak{s} \xrightarrow{*} a\varepsilon b$ . An unambiguous solution is to *create copies*  $\mathbf{U}$  and  $\mathbf{C}$  of the

original grammar variables  $\mathbf{B}$  and  $\mathbf{R}$ , respectively, so that  $\mathbf{T} \rightarrow \mathbf{T}_f | \mathbf{A}_f \mathbf{R}_f \mathbf{U}_f$  is the replacement and we add new rules  $\mathbf{U} \rightarrow \mathbf{UC} | \varepsilon$ ,  $\mathbf{C}_f \rightarrow c\mathbf{C}$ , and  $\mathbf{C}_\$ \rightarrow \varepsilon$ . The details are left to the reader.

We conclude this section with the example of Bridson and Gilman [2] alluded to in our introduction. They derive words that encode the *cutting sequence* of the line segment from the origin to each integer lattice point in the Euclidean plane as the segment crosses the horizontal ( $h$ ) and vertical ( $v$ ) lines that join lattice points. Such sequences are made unique by declaring that as a segment passes through a lattice point, the corresponding cutting sequence adds  $hv$ . For instance, the cutting sequence for the segment ending at  $(2, 4)$  is the word  $hhvvhv$ .

**Example 14.** The grammar is given by

$$\begin{aligned} \mathbf{S} &\rightarrow \mathbf{T}_\$ & \mathbf{T} &\rightarrow \mathbf{T}_q | \mathbf{T}_r | \mathbf{U}_q & \mathbf{U} &\rightarrow \mathbf{VU} | \mathbf{V} & \mathbf{V}_q &\rightarrow \mathbf{HV} & \mathbf{V}_r &\rightarrow \mathbf{V} \\ & & \mathbf{V}_\$ &\rightarrow v & \mathbf{H}_q &\rightarrow \mathbf{H} & \mathbf{H}_r &\rightarrow \mathbf{VH} & \mathbf{H}_\$ &\rightarrow h. \end{aligned}$$

Attempting to solve the grammar equations immediately runs into difficulty. The valid sentential forms  $\mathbf{V}_{qqrq\$}$  and  $\mathbf{V}_{qqqr\$}$  produce words of length eight and seven respectively, which disallows the idea of simplification via Parikh vectors. Indeed, a brute-force numerical attempt using the DSV method has exponential time complexity in the length of index strings.

We circumvent this problem by introducing two commuting formal variables  $x, y$ . Define  $L(x, y) = \sum_{i,j \geq 0} L_{i,j} x^i y^j$  where  $L_{i,j}$  counts the number of cutting sequence words that have  $i$  many occurrences of  $v$  and  $j$  many  $h$ 's. The coefficients  $L_{i,j}$  comprise a frequency distribution on the first quadrant of the integer lattice. This distribution contains more information than the one dimensional generating function  $S(z)$ . On the other hand, we can recover  $S(z) = \sum_{n \geq 1} v_n z^n$  by the formula  $v_n = \sum_{i+j=n} L_{i,j}$ .

To compute the  $L_{i,j}$ , let us simplify the grammar by ignoring the context-free copying productions  $\mathbf{U} \rightarrow \mathbf{VU} | \mathbf{V}$ , change the loading productions to  $\mathbf{T} \rightarrow \mathbf{T}_q | \mathbf{T}_r | \mathbf{V}_q$ , and begin by unloading sentential forms  $\mathbf{V}_{q\sigma}$ , where  $\sigma \in (q|r)^* \$$ . As per the proof of Lemma 3 we define a *step* as the application of the leftmost stack symbol to all non-terminals in a sentential form. If we start with an index string attached to  $\mathbf{V}$  of length  $l$ , then after  $n$  steps each non-terminal will have the same index string of length  $l - n$ . For instance if we start with  $\mathbf{V}_{qqr\sigma}$  then the first step is  $\mathbf{V}_{qqr\sigma} \rightarrow \mathbf{H}_{qr\sigma} \mathbf{V}_{qr\sigma}$ . The second step comprises two productions and ends with  $\mathbf{H}_{r\sigma} \mathbf{H}_{r\sigma} \mathbf{V}_{r\sigma}$  while the third step unloads the  $r$  from each index and results in  $\mathbf{V}_\sigma \mathbf{H}_\sigma \mathbf{V}_\sigma \mathbf{H}_\sigma \mathbf{V}_\sigma$ .

Let  $x_i$  be the number of  $\mathbf{V}$  non-terminals after performing step  $i$ , and let  $y_i$  be the number of  $\mathbf{H}$  non-terminals after performing step  $i$ . For a step unloading  $q$  we observe the recursions  $y_i = y_{i-1} + x_{i-1}$  and  $x_i = x_{i-1}$ . Likewise for  $r$  we see recursions  $y_i = y_{i-1}$  and  $x_i = y_{i-1} + x_{i-1}$ . Our simplified grammar always begins the unloading stage with  $\mathbf{V}_{q\sigma}$  and thus we obtain the initial condition  $x_1 = 1 = y_1$  regardless of  $\sigma$ . This condition, along with our recursions above imply that for each  $i$ , the pair of integers  $(x_i, y_i)$  are relatively prime.

Suppose that  $n$  is the last step needed to produce a cutting sequence word from  $\mathbf{V}_{q\sigma}$ . Identify each pair  $(x_n, y_n)$  with the corresponding point in the integer lattice, so  $x_n$  is the total number of vertical lines crossed in the cutting sequence and  $y_n$  is the number of horizontal lines crossed. (Note that  $x_n$  and  $y_n$  depend on  $\sigma$  as well as  $n$ .)

We saw above that  $\mathbf{V}_{qqr\$} \xrightarrow{*} \mathbf{V}_\$ \mathbf{H}_\$ \mathbf{V}_\$ \mathbf{H}_\$ \mathbf{V}_\$ \xrightarrow{*} v h v h v$  which is represented by  $(3, 2)$ . The pair  $(2, 3)$  is realized from  $\mathbf{V}_{qrq\$} \xrightarrow{*} h v h h v$ . Indeed, the symmetry of the grammar implies that every generated pair  $(x_n, y_n)$  has a generated mirror image  $(y_n, x_n)$  obtained by transposing each  $q$  and  $r$  in the index substring  $\sigma$  attached to the initial unloading symbol  $\mathbf{V}_{q\sigma}$ .

We claim that every relatively prime pair of positive integers is realized as  $(x_n, y_n)$  for some cutting sequence word generated by our simplified grammar. We show this by running in reverse the algorithm that generates cutting sequences. Let  $(i, j) \neq (1, 1)$  denote a coprime pair and suppose by induction that all other relatively prime pairs  $(k, l)$  are the result of unique cutting sequence words whenever  $(k < i$  and  $l \leq j)$  or  $(k \leq i$  and  $l < j)$ , *i.e.* whenever the point  $(k, l)$  is strictly below or left of the point  $(i, j)$ . If  $i < j$  then the letter  $q$  was applied at the most recent step with the previous pair being defined by  $(i, j - i)$ . On the other hand, if  $i > j$  then the rightmost letter is  $r$  and define the previous pair as  $(i - j, j)$ . In either case the new pair of coordinates remain coprime and lie in the induction hypothesis zone. Note that this is just the Euclidean algorithm for greatest common divisor and always terminates at  $(1, 1)$  when the starting pair  $(i, j)$  are coprime. Consequently the relatively prime pair  $(i, j)$  is uniquely realized as  $(x_n, y_n)$  from some cutting sequence word  $w$  generated by our simplified grammar. Furthermore  $\mathbf{V}_{q\sigma} \xrightarrow{*} w$  satisfies  $|q\sigma| = n$  which is the correct number of steps taken.

We apply our argument above to compute the two dimensional generating function  $L(x, y)$ . For our simplified grammar we have  $L_{i,j} = 1$  if the pair  $(i, j)$  is relatively prime and  $L_{i,j}$  vanishes otherwise. Equivalently,  $L_{i,j} = 1$  if and only if  $i$  and  $i + j$  are coprime. To recover  $S(z) = \sum_{n \geq 2} v_n z^n$  from  $L(x, y)$  we set  $v_n = \sum_{i+j=n} L_{i,j}$ , *i.e.* we sum along slope  $-1$  diagonal lines in quadrant one. Thus  $v_n = \varphi(n)$  where  $\varphi$  is Euler's totient function that counts the number of integers  $1 \leq i < n$  that are coprime to  $n$ . Summary: we have successfully circumvented the exponential time complexity of computing  $S(z) = \sum_{\sigma \in (q|r)^*\$} V_{q\sigma}(z)$  and found that  $S(z) = \sum_{n \geq 2} \varphi(n) z^n$ .

Recovering the original grammar by restoring the  $\mathbf{U}$  productions allows for the construction of repeated cutting sequences  $w^t$ ,  $w$  being the word associated to a coprime lattice point  $(i, j)$ . This serves to add the lattice points  $(ti, tj)$ . Here we may assume  $i$  and  $j$  are relatively prime and  $t \geq 2$  which makes these additions unique. (In fact, if  $(ti, tj) = (sk, sl)$  for another coprime pair  $(k, l)$ , then both  $s$  and  $t$  are the greatest common divisor of the ordered pair and hence  $s = t$ .) The full grammar is in bijective correspondence with the integer lattice strictly inside quadrant one. Words represent geodesics in the taxicab metric. Simple observation shows that the full growth series is represented by the rational function

$$\sum_{n \geq 2} (n-1) z^n = \frac{z^2}{(1-z)^2}$$

and as a byproduct we have re-derived Euler's identity

$$n-1 = \sum_{d|n, d>1} \varphi(d) \quad (!)$$



## 5. AMBIGUITY

We begin with the first published example of an inherently ambiguous context-free language [14, 19]. It has an unambiguous indexed grammar.

**Example 15.** Define  $\mathfrak{L}_{amb} = \{a^i b^j a^k b^l : i, j, k, l \geq 1; i = k \text{ or } j = l\}$ . The idea is to divide the language into a disjoint union of the three sub-languages

$$\mathfrak{L}_X = \{a^i b^j a^i b^l : j < l\} \quad \mathfrak{L}_Y = \{a^i b^j a^i b^l : l < j\} \quad \mathfrak{L}_Z = \{a^i b^j a^k b^j\}$$

with no restrictions on  $i, k$  other than that all exponents are at least one. An indexed grammar is

$$\mathbf{S} \rightarrow \mathbf{T}_g \mathfrak{s} \quad \mathbf{T} \rightarrow \mathbf{T}_g | \mathbf{U}_f | \mathbf{Z} \quad \mathbf{U} \rightarrow \mathbf{U}_f | \mathbf{X} | \mathbf{Y}$$

$$\mathbf{X} \rightarrow \mathbf{A} \mathbf{B} \mathbf{A} \mathbf{C} \quad \mathbf{Y} \rightarrow \mathbf{A} \mathbf{C} \mathbf{A} \mathbf{B} \quad \mathbf{Z} \rightarrow \mathbf{D} \mathbf{B} \mathbf{E} \mathbf{B}$$

$$\mathbf{A}_f \rightarrow a \mathbf{A} \quad \mathbf{A}_g \rightarrow \varepsilon \quad \mathbf{B}_f \rightarrow \mathbf{B} \quad \mathbf{B}_g \rightarrow b \mathbf{B} \quad \mathbf{B}_\mathfrak{s} \rightarrow \varepsilon$$

$$\mathbf{C}_f \rightarrow \mathbf{C} \quad \mathbf{C}_g \rightarrow b \mathbf{C} \quad \mathbf{C}_\mathfrak{s} \rightarrow b \mathbf{C}_\mathfrak{s} | b \quad \mathbf{D} \rightarrow a \mathbf{D} | a \quad \mathbf{E} \rightarrow a \mathbf{E} | a$$

The reader is invited to draw the typical parse tree and verify that the growth of this language is  $\frac{z^4(1+3z)}{(1-z)^3(1+z)^2}$ .

Several other inherently ambiguous context-free language can be generated unambiguously by an indexed grammar.

**Exercise 16.** Another early example of an inherently ambiguous context-free language is featured in [4]:  $\mathfrak{L} = \{a^n b^m c^p : m = n > 0 \text{ or } m = p > 0\}$ . Write an unambiguous indexed grammar for it. Again, split the language into the disjoint union of three types of words and build a grammar for each. The types are  $a^n b^n c^p$  with  $0 \leq p < n$ ,  $a^n b^n c^p$  with  $0 < n < p$ , and  $a^n b^p c^p$  with  $p > 0$ .

Our examples beg the question: are there inherently ambiguous indexed languages? Consider Crestin's language of palindrome pairs defined by  $\mathfrak{L}_{Crestin} = \{vw : v, w \in (a|b)^*, v = v^R, w = w^R\}$ . It is a "worst case" example of an inherently ambiguous context-free language (see [7] and its references). We conjecture that  $\mathfrak{L}_{Crestin}$  remains inherently ambiguous as an indexed language. What about inherently ambiguous languages that are not context-free?

Consider the composite numbers written in unary as per Example 13. What would an unambiguous grammar for  $\mathfrak{L}_{comp}$  look like? We would need a unique factorization for each composite  $c$ . Since the arithmetic that indexed grammars can simulate on unary output is discrete math (like addition and multiplication, no division or roots, etc), we need the Fundamental Theorem of Arithmetic. In fact, suppose there is a different unique factorization scheme for the composites, that doesn't involve a certain prime  $p$ . Then composite  $c_2 = p^2$  has only the factorization  $1 \cdot c_2$ , and similarly  $c_3 = p^3$  has unique factorization  $1 \cdot c_3$  since  $p \cdot c_2$  is disallowed. But then  $p^6 = c_2 \cdot c_2 \cdot c_2 = c_3 \cdot c_3$  has no unique factorization. Therefore all primes  $p$  are needed for any unique factorization of the set of composites. Adding any other building blocks to the set of primes ruins unique factorization.

Suppose we have an unambiguous indexed grammar for  $\mathfrak{L}_{comp}$ . It would be able to generate  $0^{p^k}$  for any prime  $p$  and all  $k > 1$ . This requires a copying mechanism

(in the manner of  $\mathbf{R}$  in Examples 7 and 13) and an encoding of  $p$  into an index string (recall our slogan “encode first, then copy”). In other words, our supposed grammar for  $\mathfrak{L}_{comp}$  must be able to first produce its complement  $\mathfrak{L}_{prime}$  and *encode these primes into index strings*. However, [18] show that the set of index strings associated to a non-terminal in an indexed grammar is necessarily a regular language. We find it highly unlikely that an indexed grammar can decode all the primes from a regular set of index strings. We conjecture that  $\mathfrak{L}_{comp} = \{0^c : c \text{ is composite}\}$  is inherently ambiguous as an indexed language.

Recall that a word is *primitive* if it is not a power of another word. In the copious literature on the subject it is customary to let  $\mathcal{Q}$  denote the language of primitive words over a two letter alphabet. It is known that  $\mathcal{Q}$  is not unambiguously context-free (see [20, 21], who exploits the original Chomsky-Schützenberger theorem listed in Section 2 above). It is a widely believed conjecture that  $\mathcal{Q}$  is not context-free at all (see [6]).

$\mathfrak{L}' = \{w^k : w \in (a|b)^*, k > 1\}$  defines the complement of  $\mathcal{Q}$  with respect to the free monoid  $(a|b)^*$ . It is not difficult to construct an ambiguous balanced grammar for  $\mathfrak{L}'$  (a simple modification of Example 11 will suffice). What about an unambiguous grammar? Recall from [17] that  $w_1^n = w_2^m$  implies that each  $w_i$  is a power of a common word  $v$ . Thus to avoid ambiguity, each building block  $w$  used to construct  $\mathfrak{L}'$  needs to be primitive. This means we must not only be able to *recreate*  $\mathcal{Q}$  in order to generate  $\mathfrak{L}'$  unambiguously, we must be able to *encode* each word  $w \in \mathcal{Q}$  as a string of index symbols, as per the language of composites. Again we find this highly unlikely and we conjecture that  $\mathfrak{L}' = \{w^k : w \in (a|b)^*, k > 1\}$  is inherently ambiguous as an indexed language.

## 6. OPEN QUESTIONS

We observe that the generating function  $S(z)$  of an indexed language is an infinite sum (or multiple sums) of a family of functions related by a finite depth recursion (or products/sums of the same). Into what class do the generating functions of indexed languages fit?

Is Crestin’s language inherently ambiguous as an indexed language? What about the composite numbers in unary or the complement of the primitive words?

We end where we began with the non-context-free language  $\{a^n b^n c^n : n > 0\}$ . It has a context-sensitive grammar

$$\mathbf{S} \rightarrow abc|a\mathbf{B}Sc \quad \mathbf{B}a \rightarrow a\mathbf{B} \quad b\mathbf{B} \rightarrow bb$$

for which the original DSV method works perfectly. The method fails for several other grammars generating this same language. What are the necessary and sufficient conditions to extend the method to generic context-sensitive grammars?

## ACKNOWLEDGEMENTS

Daniel Jorgensen contributed ideas to the first two authors. The third author offers thanks to Mike Zabrocki and the participants of the Algebraic Combinatorics seminar at the Fields Institute for Research in Mathematical Science (Toronto, Canada) for early discussions on this theme.

## REFERENCES

- [1] Alfred V. Aho. Indexed grammars—an extension of context-free grammars. *J. Assoc. Comput. Mach.*, **15**:647–671, 1968.
- [2] Martin R. Bridson and Robert H. Gilman. Formal language theory and the geometry of 3-manifolds. *Comment. Math. Helv.*, **71** (4):525–555, 1996.
- [3] Martin R. Bridson and Robert H. Gilman. Context-free languages of sub-exponential growth. *J. Comput. System Sci.*, **64** (2):308–310, 2002.
- [4] N. Chomsky and M. P. Schützenberger. The algebraic theory of context-free languages. In *Computer programming and formal systems*, pages 118–161. North-Holland, Amsterdam, 1963.
- [5] Delest M. Algebraic languages: a bridge between combinatorics and computer science. In *Formal power series and algebraic combinatorics (New Brunswick, NJ, 1994)*, pages 7187. DIMACS Ser. Discrete Math. Theoret. Comput. Sci., **24**, Amer. Math. Soc., Providence, RI, 1996.
- [6] Pál Dömösi, Sándor Horváth, Masami Ito, László Kászonyi, and Masashi Katsura. Some combinatorial properties of words, and the Chomsky-hierarchy. In *Words, languages and combinatorics, II (Kyoto, 1992)*, pages 105–123. World Sci. Publ., River Edge, NJ, 1994.
- [7] Philippe Flajolet. Analytic models and ambiguity of context-free languages. *Theoret. Comput. Sci.*, **49** (2-3):283–309, 1987. Twelfth international colloquium on automata, languages and programming (Nafplion, 1985).
- [8] Philippe Flajolet and Robert Sedgewick. *Analytic combinatorics*. Cambridge University Press, Cambridge, 2009.
- [9] Eric M. Freden and Jennifer Schofield. The growth series for Higman 3. *J. Group Theory*, **11** (2):277–298, 2008.
- [10] Robert H. Gilman. A shrinking lemma for indexed languages. *Theoret. Comput. Sci.*, **163** (1-2):277–281, 1996.
- [11] Robert H. Gilman. Formal languages and their application to combinatorial group theory. In *Groups, languages, algorithms*, volume 378 of *Contemp. Math.*, pages 1–36. Amer. Math. Soc., Providence, RI, 2005.
- [12] R. I. Grigorchuk and A. Machì. An example of an indexed language of intermediate growth. *Theoret. Comput. Sci.*, **215** (1-2):325–327, 1999.
- [13] Derek F. Holt and Claas E. Röver. Groups with indexed co-word problem. *Internat. J. Algebra Comput.*, **16** (5):985–1014, 2006.
- [14] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley Publishing Co., Reading, Mass., 1979. Addison-Wesley Series in Computer Science.
- [15] Roberto Incitti. The growth function of context-free languages. *Theoret. Comput. Sci.*, **255** (1-2):601–605, 2001.
- [16] Leonard Lipschitz and Lee A. Rubel. A gap theorem for power series solutions of algebraic differential equations *Amer. J. Math.*, **108** (5):1193–1213, 1986.
- [17] R. C. Lyndon and M. P. Schützenberger. The equation  $a^M = b^N c^P$  in a free group. *Michigan Math. J.*, **9**:289–298, 1962.
- [18] R. Parchmann and J. Duske. The structure of index sets and reduced indexed grammars. *RAIRO Inform. Théor. Appl.*, **24** (1):89–104, 1990.
- [19] R. J. Parikh. On context-free languages. *J. of the ACM*, **13**:570–581, 1966.
- [20] H. Petersen. The ambiguity of primitive words. In *STACS 94 (Caen, 1994)*, volume 775 of *Lecture Notes in Comput. Sci.*, pages 679–690. Springer, Berlin, 1994.
- [21] H. Petersen. On the language of primitive words. *Theoret. Comput. Sci.*, **161** (1-2):141–156, 1996.
- [22] Sarah Rees. A language theoretic analysis of combings. In *Groups, languages and geometry (South Hadley, MA, 1998)*, volume 250 of *Contemp. Math.*, pages 117–136. Amer. Math. Soc., Providence, RI, 1999.

DEPARTMENT OF MATHEMATICS, SOUTHERN UTAH UNIVERSITY, CEDAR CITY, UT, USA 84720

DEPARTMENT OF MATHEMATICS, SOUTHERN UTAH UNIVERSITY, CEDAR CITY, UT, USA 84720

DEPARTMENT OF MATHEMATICS, SIMON FRASER UNIVERSITY, BURNABY BC, CANADA V5A 1S6